XO: XMPP Overlay Service for Distributed Chat

Robert N. Lass*, Joe Macker† David Millar*, William C. Regli* and Ian Taylor‡
*Department of Computer Science
College of Engineering
Drexel University, Philadelphia, PA
Email: {urlass, dwm27, regli}@cs.drexel.edu

†Information Technology Division
Naval Research Lab, Washington, D.C.
Email: joseph.macker@nrl.navy.mil

‡School of Computer Science
Cardiff University, Cardiff, UK
Email: Ian.J.Taylor@cs.cardiff.ac.uk

***Abstract*—This work discusses the adaptation of group-oriented messaging and chat technology for operation in serverless, multicast-capable mobile wireless architectures. The main goals are to allow group messaging and chat sessions to fragment/coalesce, operate through disrupted TCP conditions, and improve bandwidth utilization when multicast is available. In addition, the solution demonstrates proxying and gateway methods to interoperate with existing client and server standards and software. The approach presents several innovations that extend and adapt eXtensible Messaging and Presence Protocol (XMPP) standards for incorporating group serverless chat and messaging within more challenging operational environments. While there is large body of work on messaging middleware solutions, this paper concentrates on the adaptation of specific XMPP standards for serverless, multicast operation. We discuss our working implementation prototype and present initial experimentation comparing client/server multi-user chat (MUC) operation to serverless multicast MUC within several mobile network scenarios. In addition, we demonstrate a gatewaying solution for serverless MUC systems to interoperate with conventional MUC server-based systems. The specific test scenarios are instrumented to operate within a wireless mobile emulation environment using mobile ad hoc network (MANET) unicast and multicast routing technology. This approach remains independent of any particular routing algorithm and the proxied XMPP protocol module allows for the deployment of existing real-world client software across all nodes of the network. The initial findings show the significant performance potential for serverless MUC extensions. In addition to these results, we discuss some ongoing design challenges and future planned work.**

I. Introduction

Presently a wide variety of chat and messaging systems exist and are deployed for multi-user chat (MUC) capabilities (e.g., Internet Relay Chat (IRC), eXtensible Messaging and Presence Protocol (XMPP) jabber, gchat). These widely used systems generally have salient features making performance problematic in wireless mesh or mobile ad hoc network (MANET) scenarios. First, the standard architectures are often purely client/server based designs and if the MUC management servers disappear, degrade, or fragment, the client group suffers with potentially no communication support even if peer collaborative nodes are reliably reachable within the network. Second, present approaches tend to build and rely upon establishing multiple TCP unicast transport session for maintaining centralized MUC chat, raising mobile and wireless performance issues [1], [2]. In addition, supporting a set of centralized unicast connections for group communications is resource inefficient in cases where the flows traverse shared wireless medium. An example of a unicast-oriented, centralized MUC standard is the XMPP MUC protocol [3] (client/server protocol), described in more detail in Section II. To begin addressing the above concerns while maintaining messaging standard interoperability, we have developed proxying and gateway components that leverage existing XMPP-based MUC messaging standards to provide a basic multicast and distributed peer-to-peer operational mode. This approach is essentially an XMPP Overlay (XO) messaging service that adapts XMPP MUC to a mobile multicast environment and provides alternative network transport capabilities. Since XO provides serverless operation and there is no centralized repository, we also developed additional multi-hop discovery and presence mechanisms to improve mobility robustness and to aid in simplifying network operations.

In addition to demonstrating novel multicast, serverless capabilities enabled by XO, we are interested in supporting existing widely available client and server software for XMPP MUC simultaneously. Nodes using the serverless paradigm and those using the conventional centralized MUC can also interoperate in bridged group chat/messaging sessions. Local XMPP clients interact with a proxy and these localized sessions can interoperate with conventional XMPP servers when gatewaying is available. If gatewaying is not available, or becomes disrupted, XO can operate as a completely serverless MUC in an independent, autonomous fashion. To summarize, the solution space addresses the following components:

- multicast serverless chat capability;

978-1-4244-8179-8/10/$26.00 ©2010 IEEE

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
NOV 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
XO: XMPP Overlay Service for Distributed Chat	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Naval Research Laboratory,Information Technology Division,Washington,DC,20375	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

Presented at MILCOM 2010 Military Communications Conference, Oct 31, Nov 3, 2010, San Jose, CA

14. ABSTRACT

This work discusses the adaptation of grouporiented messaging and chat technology for operation in serverless multicast-capable mobile wireless architectures. The main goals are to allow group messaging and chat sessions to fragment/ coalesce, operate through disrupted TCP conditions, and improve bandwidth utilization when multicast is available. In addition, the solution demonstrates proxying and gateway methods to interoperate with existing client and server standards and software. The approach presents several innovations that extend and adapt eXtensible Messaging and Presence Protocol (XMPP) standards for incorporating group serverless chat and messaging within more challenging operational environments. While there is large body of work on messaging middleware solutions, this paper concentrates on the adaptation of specific XMPP standards for serverless, multicast operation. We discuss our working implementation prototype and present initial experimentation comparing client/server multi-user chat (MUC) operation to serverless multicast MUC within several mobile network scenarios. In addition, we demonstrate a gatewaying solution for serverless MUC systems to interoperate with conventional MUC server-based systems. The specific test scenarios are instrumented to operate within a wireless mobile emulation environment using mobile ad hoc network (MANET) unicast and multicast routing technology. This approach remains independent of any particular routing algorithm and the proxied XMPP protocol module allows for the deployment of existing real-world client software across all nodes of the network. The initial findings show the significant performance potential for serverless MUC extensions. In addition to these results, we discuss some ongoing design challenges and future planned work.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**6**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

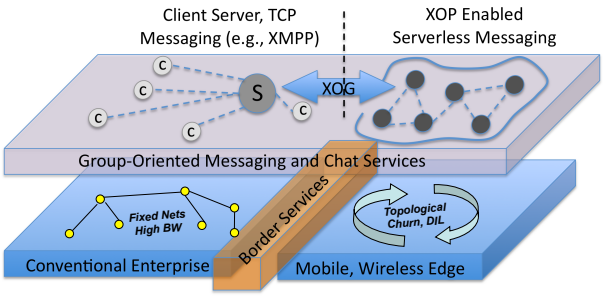


Fig. 1. Problem Space

- multi-hop serverless discovery support;
- standardized XMPP MUC client proxying; and
- standardized XMPP MUC server gatewaying.

The main contributions of this paper are the architecture of XO and the initial empirical analysis of the multicast-based XMPP Overlay (XO) service. The XO design provides both a proxy and a server gateway component as represented in Figure 1. An unmodified XMPP client connects locally to the XO proxy (XOP) agent, which mediates the XMPP protocol to XO multicast message forwarding and related discovery processes. Finally, the XO Gateway (XOG) bridges XOP MUC multicast operation with conventional XMPP MUC sessions using off-the-shelf client/server solutions (e.g., Openfire[1] server).

The paper is organized as follows. Section II provides an overview of related XMPP Extension Protocols (XEPs) and multicast work. Section III discusses the architecture, design and system components for the XO system. Section IV discusses various empirical studies conducted using XO components. Section V presents conclusions and lessons learned and Section VI outlines future work and ongoing challenges.

II. Related XMPP Standards and Related Multicast Work

This section discussed related XMPP standards and the relationship to XO work and also briefly summarizes related multicast work that XO takes advantage of. Due to space limitations and the depth of the field, this is not a comprehensive survey of related work but rather it concentrates on the specifics of adapting XMPP and analyzing XO.

A. XMPP Extensions

XEP-0174 [4] is a Draft Standard of the XMPP Standards Foundation that specifies a link-local messaging protocol defining how XMPP messaging can be accomplished using zero-configuration networking. This method uses mDNS for service discovery of network entities that support the protocol, including their IP addresses and preferred ports. Any two entities can then negotiate a serverless connection and using XML streams, exchange XMPP messages and IQ stanzas. XEP-0174 is similar to XO in that it supports serverless discovery and chat, but it only supports link-local discovery, and point-to-point messaging over TCP streams.

[1]http://www.igniterealtime.org/projects/openfire/index.jsp

XEP-0100 [5] specifies XMPP gateway interaction. The specification defines this to mean gateways that proxy XMPP clients onto non-XMPP servers, such as IRC. This is similar to what is occurring with the XOP, in that it receives XMPP packets and translates them into another protocol. However, there are two main differences. First, the XEP requires a client that implements the XEP registering with a gateway, most likely on an XMPP server such as OpenFire, that has also implemented the XEP. XOP does not require a client or server with these features. Second, the XEP does not support MUC, only point-to-point messaging.

XEP-0045 [3] describes the XMPP Multi-user chat (MUC) protocol. This protocol allows clients to create, discover, join and leave group-oriented MUC rooms. The rooms are associated with a server, and all messages addressed to the room are sent to the server. The server then resends the messages out to all of the members of the room through their existing connection to the server. The main difference between this and XO is the requirement of the server, and the multiple server-client unicast connections required versus the multicast approach to transporting messages among peers.

Among the XMPP extensions XEP-0100, XEP-0174, and XEP-0045, none address how the TCP-oriented connection paradigm might apply to a multicast setting and therefore do not offer serverless group-based messaging. XEP-0174 specifies serverless discovery over a link-local connection, but then delegates all subsequent interactions to a unicast TCP connection. With the other two extensions, all communications pass through an XMPP server which multiplexes messages over TCP connections with each of the various chat room members. XO is capable of not only initiating an XMPP session, but it can also utilize underlying out-of-band protocols for the communication of actual XMPP stanzas, such as chat messages and so forth.

B. Related Multicast Work

The work on XO in this paper is partially built on the idea that some form of multi-hop multicast forwarding capability is available within wireless edge networks. If this is not the case, a deployment may take advantage of XO serverless, non-TCP transport adaptations to XMPP but our main contribution is in the multicast space. Future work is looking at alternative pluggable transport capabilities for more survivable, consistent operations even in unicast scenarios.

To support the experimental scenarios in this paper, XO takes advantage of the Simplified Multicast Forwarding mechanism [6] to provide an IP multicast forwarding interface within mobile mesh or MANET wireless environments. SMF is capable of supporting multiple forwarding algorithms such as classical flooding, essential connecting dominating set (E-CDS) and source-based multipoint relay (S-MPR). Other IP or MANET specific multicast protocols could be used with no change to the XO approach.

III. System Components and Design

XO is implemented as a plugin to the Generic Unicast-to-Multicast (GUMP) proxy. GUMP is a framework allowing

developers to create transport protocol independent server-side proxies that support the conversion of messaging protocols, such as XMPP, into a back-end serverless group messaging environments. GUMP allows the developer to focus on the detail of the server proxy without being tied down to the specifics of how entities are discovered nor how messages are to be sent between those entities e.g. multicast. The GUMP architecture is divided into four main areas: application connectivity; server proxy interface and implementation; session messaging; and data messaging.

Application connectivity addresses how applications connect to GUMP. This uses a TCP input binding, which exposes a TCP server for connection from a standard XMPP application. Other input schemes are available however, such as HTTP and native Java connectivity, through a GUMP java.net socket factory implementation. The *Server Proxy* provides a pluggable interface to multiple messaging protocol server proxies, each addressing a specific protocol e.g. XMPP, WS-Notification and so forth. **Session Messaging** provides an interface to various discovery subsystems (e.g. multicast DNS for use by the proxy to be able to advertise and subsequently discover entities on the network. Finally, **Data Messaging** provides an interface to various underlying multicast implementations. Primarily, this work focuses on investigating two implementations here: the default Java multicast implementation and NORM (NACK-Oriented Reliable Multicast) [7] transport for capabilities requiring reliable multicast delivery.

A. XMPP Proxy, Session and Data Messaging

One of the key features of XO is that it can be used with existing XMPP clients without any modification. The proxy is typically run on the same node as the XMPP clients and a client opens a TCP connection with the proxy on port 5222, and authenticates. The current version of XO does not implement security, and therefore the proxy authenticates any username/password pair. However, XO accepts authentication messages and this can be extended to implement the desired behavior in the future. Once the socket has been created, a client sets up a stream the same as it would with a server. The proxy then processes any *presence*, *iq* or *message* messages it receives from the client until the stream is closed.

When the proxy receives a *message* or *presence* packet from the client, it passes it to the packet router. The packet router is an application level XMPP stanza/packet router, which routes incoming packets to their representative software endpoints. Examples of endpoints include users locally logged into the XOP instance, a MUC component or other such component. When entities connect to the XOP, the packet router stores routes to them. The packet router determines if the destination is local (a connected entity) or remote. If it is local, it forwards it to the connected entity, otherwise it forwards it to the GUMP multicast interface. Finally, GUMP sends the message out using whatever protocol (UDP multicast, NORM, etc) it is currently configured to use.

For receiving packets, XOP uses a GUMP multicast socket. Incoming packets are sent to the packet router, which then forwards them to the appropriate connected entity, if any. The transport mechanism for incoming packets is determined by the GUMP configuration, the XMPP component responsible for listening for packets on the network has no need to know the specific protocol being used.

When the proxy receives an *iq* (info/query) packet from the client, it again passes it to the packet router. If it is local, it is forwarded to the connected entity, otherwise it passes it to the IQManager component of the XMPP proxy, which updates a context object if necessary. The context object is intended to help manage the state of the proxy. Keep in mind that with no server, some of the state needs to be managed by the proxy. The context is not used at this point, but could be used in the future, for example to cache request / response pairs or to enforce security/policy issues. Finally, the IQManager performs the appropriate action (advertise service, reply to the sender with "server information", query for available services, etc) using GUMP's discovery system. GUMP then performs whatever action is needed for the currently configured discovery protocol (e.g., multi-hop mDNS in our experiment case), and returns an appropriate response to the client if appropriate. Specifically, the XOP registers a listener with the GUMP discovery interface when it starts up. When a discovery-related event occurs, the XOP creates an *iq* packet and forwards it to the packet router for delivery to the appropriate entity, or updates the state e.g. adds a MUC room to the available rooms.

For *topic processing* and to avoid the potential future issues involved with sending all topics to the same multicast address, the topic names are hashed to a range of multicast addresses in the IPv4 organization local scope (*i.e.* the `239.192.0.0./14` subnet). The exact process is to take an MD5 hash of the full Jabber ID (the identifier used by XMPP of the form [node "@"] domain ["/" resource]) of the room name (*e.g.*: room@conference.proxy). Then, XO performs a logical OR on the last 18 significant bits of the hash and `239.192.0.0` to calculate the multicast address. Any collisions are handled by the receiver filtering against the room name. There are longer term issues with standardizing multicast topic rooms assignment or management but this initial approach is taken as a working "proof of concept." Multiple additional or alternative capabilities can be easily engineered.

B. Gatewaying

As mentioned in the introduction, if the proxy is running on a node with connections to both a multicast MANET and a conventional enterprise environment, it can be setup as a gateway using XOG which acts as a bridge between the peer-to-peer XO multicast protocol and the conventional client-to-server XMPP-MUC protocol. The XOG establishes a connection with an XMPP MUC server and translates packets between XO and conventional XMPP. If a packet arrives on the GUMP discovery interface, the XOG translates it into an *iq* packet and sends it to the receiving server. Likewise, if it receives a presence or message packet from the GUMP multicast interface it forwards it. Vice versa, when an XMPP packet arrives from the receiving server, either the appropriate functions are called in the GUMP discovery interface, or the

packets are converted to XO and handed to the GUMP multicast interface. This service allows XO sessions on dynamic mesh and mobile networks to participate in standard XMPP MUC sessions on other network environments.

Three designs were considered for Gatewaying:

- Use one TCP socket for each client being proxied;
- Use a custom server plugin; or
- Use the standard server-to-server protocol ("Server Dialback").

Using one TCP socket for each client being proxied has the advantage of being easy to implement using existing XMPP open source libraries. In this option, the proxy opens one TCP socket per client being proxied, as though it were the client. No modification to the server would be necessary. The disadvantage of this is that it may require an excessive resources to maintain sockets for each client being proxied. Another solution to reduce the number of TCP sessions being maintained is to create a server plugin to handle communications with the proxy. The main disadvantage here is that it will not work with unmodified XMPP servers and that it cannot work with server software other than that for which the plugin was written. A third solution is to use XMPP's server dialback protocol (DB), which is used for server-to-server communications. The protocol opens a TCP socket, authenticates, and then may send *message*, *presence*, and *iq* messages over the socket. This could also be used to gateway between two XO clusters. We have a preliminary implementation of the DB approach that supports one XOG per XO group, but there is no limit on the number XO groups that could connect to a server.

IV. Empirical Mobile Network Studies using XO

All of the experiments described in this work were performed on the Common Open Research Emulator (CORE) environment running on FreeBSD. CORE [8] allows for low-fidelity emulation of wired and wireless networks between virtual machines. These virtual machines can be distributed across multiple machines running CORE, and also connected to real non-virtualized network interfaces. In these experiments, all of the hosts were emulated inside CORE.

The scenarios used in our basic experiments were notional representation of platoon level mobility over an area. Mobile entities continuously communicate XMPP message stanzas and additional text chats amongst mobile peer network nodes – see Figure 2. A testbed was also established to demonstrate that the MUC sessions can also be back-hauled to a centralized MUC server over an emulated long-haul wireless link using XOG. In the scenario the backhaul connection can be removed and platoon level network mobility allows for fragmentation to demonstrate the ability to maintain autonomous messaging locally when operating in a disconnected manner. To study statistically the performance message delivery an XMPP MUC traffic generator was used to automate evaluations of distributed MUC performance. In our experiment, each node uses a traffic generator to send traffic, which is written using the Ignite Realtime Smack library [2], and therefore can send traffic

[2]http://www.igniterealtime.org/projects/smack/

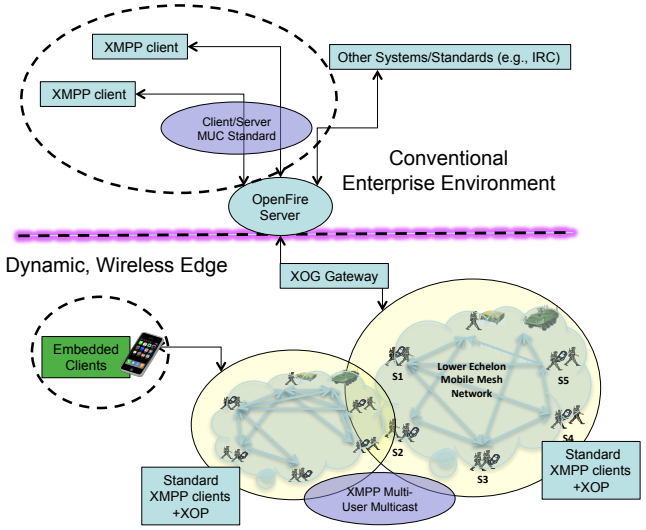


Fig. 2. Testbed Demonstration: Tactical nodes chatting, with a backhaul link to a standard enterprise XMPP deployment.

to a room with or without the XO proxy.

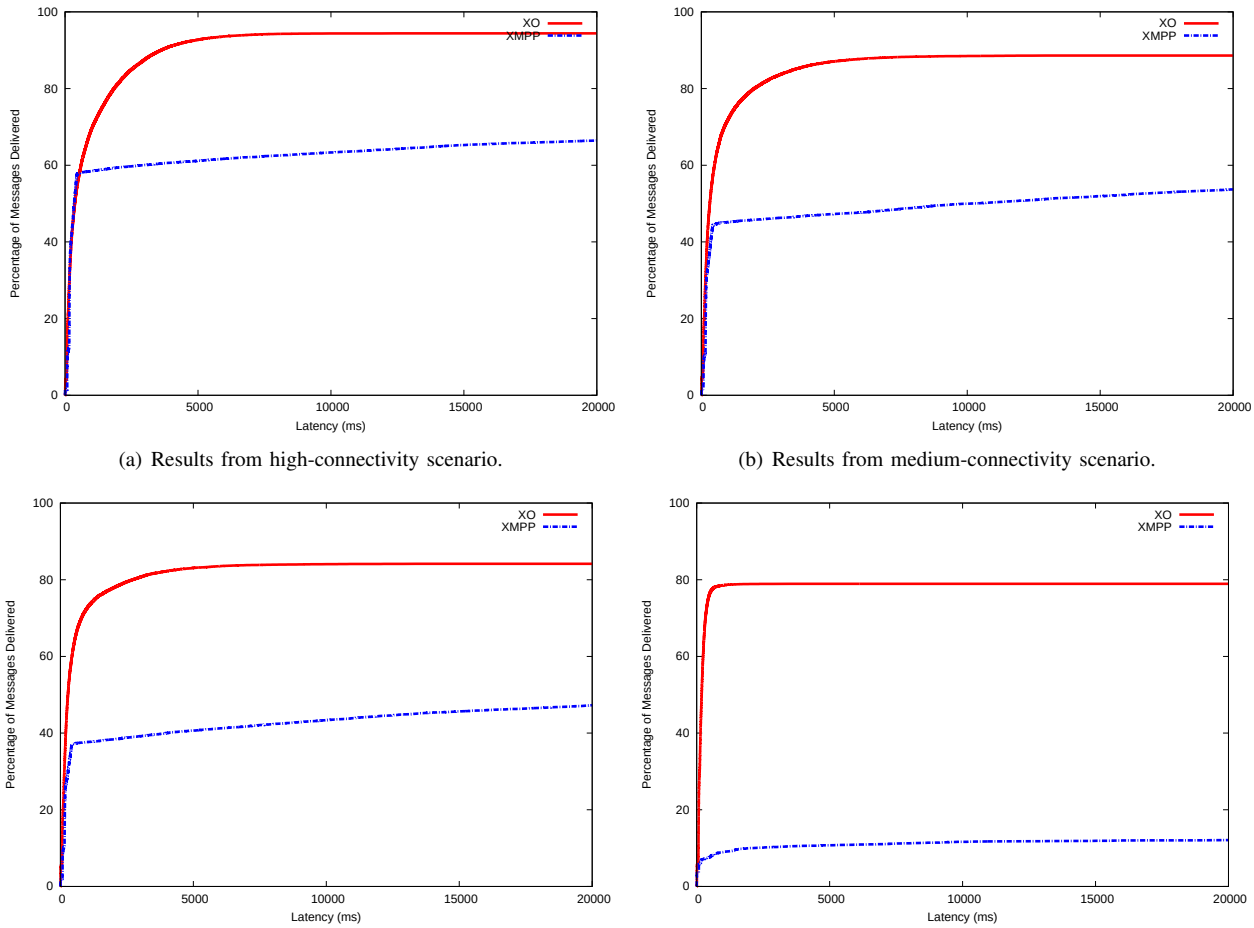
The nodes in the platoon move based on the reference point group mobility model (RPGM), using the Colorado School of Mines mobility generation tool [9]. This model is a generalized case of the most commonly used group mobility models. The parameters used in these experiments appear below:

Parameter	Value
Number of Groups	1
Nodes per Group	11
Reference Point (RP) Separation	50
Node Separation from RP	varied
Max X	1800
Max Y	1800
End Time	300
Speed Mean	30
Speed Delta	5
Pause Time	2
Pause Delta	2

The only parameter varied was the node separation from the reference point (SRP), which allowed for nodes to roam closer or farther away from the reference point, creating different density including variable probability of fragmentation. The three values used had the following qualitative descriptions:

- **Low connectivity**: SRP was set to 400. The nodes frequently spread out into networks of four to six hops in diameter. Disconnects were common.
- **Medium connectivity**: SRP was set to 320. The network was two or three hops in diameter most of the time, and disconnects occurred less frequently.
- **High connectivity**: SRP was set to 230. The network was tightly clustered, usually within one or two hops. Disconnects were rare.

These three mobility scenarios are the scenarios used for the experiments. There is also a fourth scenario, which the same as the High Connectivity scenario except that a probability of bit error, $Pb = 10^{-5}$, was added to the emulated wireless network channel. Five instances of each of the four types was created, and one experiment using XO and one experiment



(a) Results from high-connectivity scenario.

(b) Results from medium-connectivity scenario.

(c) Results from low-connectivity scenario.

(d) Results from high-connectivity scenario with bit error.

Fig. 3. Latency results for XMPP and XO in the four scenarios. Red solid lines represent XO and blue dashed lines XMPP. The y-axis indicates the percentage of messages delivered at the corresponding time on the x-axis, truncated at 20 seconds of delay. The XMPP CDF tail continued changing well past this delay, but not significantly.

using XMPP was done with each instance. Each messages sent was one sentence from the Declaration of Independence. The intra-message sending times were exponentially distributed with $\lambda = 0.2$. Each of the ten nodes in the experiments were configured to both send and receive messages.

A. Baseline

As a baseline for comparison, all of the experiments were first run configured as XMPP clients communicating with the XMPP-MUC server in the standard manner. Each of the nodes ran the Quagga OSPF routing suite [3] to transport packets to the gateway (node with the link to the infrastructure network), which routes them to the server. This was accomplished in CORE using an RJ45 connection to a LAN with an OpenFire XMPP server running on it.

This configuration presented some problems in this environment, because the nodes become disconnected from the server at times, which often cause the TCP sockets to close. This was handled by attempting to re-open the TCP socket if it was not open and a message was ready to be sent. If that failed, it would wait one second and try again, repeating until it was successful. In the XO configuration, nothing was changed except that all of the nodes ran SMF and the XMPP traffic generators were configured to use the XO proxy rather than directly connecting with the sever.

[3]http://www.quagga.net/

B. Results

Figure 3 shows basic results from multiple scenario trials as a series of cumulative density function (CDF) graphs. The y axes represent the cumulative message delivery ratio vs. message delivery delay represented on the x axis. In all cases, XO significantly outperforms the conventional XMPP client/server approach in terms of both message delivery ratio and overall message delay. This effect is further amplified when the emulated link conditions are degraded by included statistical channel loss in addition to mobility induced dynamics. Under all scenario conditions the conventional XMPP MUC operating in MANET environments has significantly skewed message delay statistics with a long tail. In many cases, there is no message delivery to a client due to socket failures and TCP issues. We attempt to be fair to XMPP operation and provide a reasonable short term reconnect time.

While XO suffers from some message loss as expected in this dynamic environment, the collection of XO message delay CDFs demonstrates a low overall message delivery delay statistic which is highly encouraging. XO delivered more messages, and the distribution of the latencies has relatively low variance with the vast majority of the messages delivered in less than a second.

There are several caveats that should be mentioned about our comparison of XMPP and XO. The first is the technique for allowing the re-opening of TCP sockets that have closed due to network errors or failures in the XMPP case. This is certainly not the only approach one could take to mitigate the problem of closed sockets, and there may be better approaches that could be examined and optimized. However, again these modifications would not be conventional XMPP MUC instantiations and we are comparing in a relatively fair manner with present solutions. Other approaches people might take to make XMPP perform better could include running multiple servers on the tactical edge, or having multiple gateway nodes to the enterprise network. We are not claiming that the performance we have observed is indicative of the best XMPP given all architectural choices, but we think it is a valid baseline to compare against conventional client/server approaches within the given MANET scenarios.

V. Conclusions and Lessons Learned

This paper presented a new XMPP overlay (XO) service that provides serverless, multicast multi-user messaging and chat. XO extends presents XMPP standards making it possible to consider its use as a distributed chat capability within mobile, wireless network environments. XO works with both existing XMPP MUC clients and also provides basic gateway services to conventional XMPP server-based MUC deployments. This provides interoperation between more tactical edge network environment operations and stable enterprise deployment environments. This paper also described and presented initial empirical results comparing a working implementation of serverless XO against more conventional TCP-based XMPP MUC within a set of emulated mobile network scenarios varying in density. Initial results demonstrate that XO has significant advantages in terms of both message delivery ratio and message latency across the set of mobile network scenarios studied. Overall, XO service extensions are a promising approach to adapting group-wise messaging in dynamic mobile mesh networks that preserves the attractive, extensible messaging features of XMPP and allows interoperation with existing gateways and client software.

VI. Future Work

XO is presently a work in progress, with several areas undergoing further examination and development. First off, we intend to perform similar performance trials using more detailed networking models, including more accurate topological and wireless conditions. The behavior and performance of multicast forwarding is often dependent upon the lower layers of the wireless system and we are interested in examining those tradeoffs further. While XO is independent of any particular multicast approach we are also interested in which multicast protocols or modes provide robust performance for group messaging protocols such as XO, in terms of both delay and delivery ratios. Also the addition of multicast-capable reliable transport protocols, such as NORM, is possible due to the pluggable transport design approach but the proper use of such protocols for this application area remains an open research issue that we are pursuing in future efforts.

In developing serverless and multicast-based messaging operation new challenges arise related to session management and data persistence due to the distributed, dynamic operation. Data consistency or persistence across dynamic, serverless network services is an also open area for further work that we are pursuing. Since the network can be unreliable as well as fragment and merge, a mechanism for persisting data or even prioritizing the persistence mechanism is needed beyond typical reliable multicast protocols. We are now examining multiple algorithms and approaches to provide a modular persistence mechanism to supplement XO. This mechanism will address data heterogeneity along with congestion control and will allow for particular mission and data driven configurations for persistence. Techniques such as epidemic protocols or other disruption tolerant technologies might be applicable but more work is needed to understand the value and applicability in these dynamic environments. Finally, there is the issue of using XO as a general message bus (as can be done with XMPP extensions) even when not fragmented. This opens up several design issues including efficient encoding, compression, and the proper management of multiple message channels. At present we can provide a basic XMPP stanza compression scheme prior to network forwarding to improve bandwidth utilization. In general, these areas remain open for significant further study and work.

Finally, we need to examine what the difference in network overhead is in using XO versus XMPP. This could mean comparing *all* (*e.g.*, duplicate packets, control packets, etc) of the packets that are sent as a result of each chat, which would require low-level instrumentation.

References

[1] Z. Fu, P. Zerfos, H. Luo, L. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *IEEE INFOCOM*, vol. 3. Citeseer, 2003, pp. 1744–1753.

[2] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, 2003.

[3] "Multi User Chat (MUC) - XEP-0045." [Online]. Available: http://xmpp.org/extensions/xep-0045.html

[4] "Link-Local Messaging - XEP-0174." [Online]. Available: http://xmpp.org/protocols/linklocal/

[5] "Gateway Interaction - XEP-0100." [Online]. Available: http://xmpp.org/extensions/xep-0100.html

[6] J. Macker, et al, "Simplified Multicast Forwarding," Mar 2010, http://tools.ietf.org/html/draft-ietf-manet-smf-10.

[7] B. Adamson et al, "Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol." [Online]. Available: http://tools.ietf.org/html/rfc5740

[8] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *IEEE Military Communications Conference*, 2008, pp. 1–7.

[9] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.